

# Package: shrthnd (via r-universe)

January 30, 2025

**Title** Work with data that uses shorthand and symbols

**Version** 0.1.0.9000

**Description** Process character vectors of numerical data that contains non-numeric shorthand and symbols.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.3

**Imports** cli, dplyr, glue, pillar, purrr, rlang, stringr, tibble, tidy, utils, vctrs (>= 0.6.2)

**Suggests** knitr, palmerpenguins, rmarkdown, testthat (>= 3.0.0)

**URL** <https://github.com/mattkerlogue/shrthnd>,  
<https://mattkerlogue.github.io/shrthnd/>

**BugReports** <https://github.com/mattkerlogue/shrthnd/issues>

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**Config/pak/sysreqs** libicu-dev

**Repository** <https://mattkerlogue.r-universe.dev>

**RemoteUrl** <https://github.com/mattkerlogue/shrthnd>

**RemoteRef** HEAD

**RemoteSha** ca9ff1a93c9d90b8b30ba6565038649d42e0827c

## Contents

annotations . . . . .	2
as_shrthnd . . . . .	4
find_annotations . . . . .	5
is_shrthnd_num . . . . .	7
make_shrthnd_num . . . . .	8

note_to_title . . . . .	9
shrthnd_coercion . . . . .	10
shrthnd_list . . . . .	11
shrthnd_maths . . . . .	12
shrthnd_num . . . . .	13
shrthnd_tags . . . . .	15
shrthnd_tbl . . . . .	15
tag_match . . . . .	16
where_shrthnd_cols . . . . .	18
zap_shrthnd . . . . .	19

## Index 21

---

annotations	<i>Get and set annotations for a tibble</i>
-------------	---

---

### Description

A `shrthnd_tbl()` has three sets of annotations that can be defined: a `title`, a `source_note` and a set of general notes. This family of functions allows you to view and modify these notes.

### Usage

```

annotations(x)

shrthnd_title(x)

shrthnd_title(x) <- value

set_title(x, value, .overwrite = FALSE)

shrthnd_source_note(x)

shrthnd_source_note(x) <- value

set_source_note(x, value, .overwrite = FALSE)

shrthnd_notes(x)

shrthnd_notes(x) <- value

set_notes(x, value, .overwrite = FALSE)

add_notes(x, value, .add_before = Inf)

add_notes(x) <- value

set_tbl_antn(

```

```
x,
  what = c("title", "source_note", "notes"),
  value,
  .overwrite = FALSE,
  .add = FALSE,
  .add_before = Inf
)
```

### Arguments

x	A shrthnd_tbl() object
value	The value to set
.overwrite	Whether an existing value should be overwritten
.add_before	When adding notes, where to add the note (defaults to the end of the current set of notes)
what	Which note to set, one of title, source_note or notes
.add	When what = "notes", whether to append to the existing set of notes

### Details

Use annotations() to see the all the annotations associated with a shrthnd\_tbl() object.

Use shrthnd\_title(), shrthnd\_source\_note() and shrthnd\_notes() get the relevant annotations(s) of a shrthnd\_tbl() object. Passing a value to these functions (e.g. shrthnd\_title(x) <- "My title") will set the value of these annotation, overwriting the existing value(s).

set\_title(), set\_source\_note(), and set\_notes() also allow you to set the value of these annotations. By default they will not permit overwriting of existing values, setting .overwrite = TRUE permits this.

add\_notes() allows you to append notes to the existing set of general notes.

set\_tbl\_antn() is a low level helper function that powers the assignment operations.

### Value

For shrthnd\_title(), shrthnd\_source\_note() and shrthnd\_notes() a character vector of the note(s). For the setting functions returns invisibly either x if the attribute was set or NULL if not.

### See Also

[note\\_to\\_title\(\)](#), [shrthnd\\_tbl\(\)](#), [zap\\_shrthnd\(\)](#)

### Examples

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
tbl <- tibble::tibble(x = x, sh_x = sh_x)

sh_tbl <- shrthnd_tbl(tbl) |>
  set_title("My Example Table") |>
```

```

set_source_note("Shrthnd documentation (2023)") |>
set_notes(c("Note 1", "Note 2"))

sh_tbl

annotations(sh_tbl)

shrthnd_title(sh_tbl)
shrthnd_source_note(sh_tbl)
shrthnd_notes(sh_tbl)

add_notes(sh_tbl) <- "Note 3"
shrthnd_notes(sh_tbl)

```

---

as\_shrthnd

*Coerce a shrthnd\_num to a character vector with shorthand*


---

## Description

as\_shrthnd() coerces a shrthnd\_num() vector back to a character vector re-inserting the shorthand tags.

## Usage

```

as_shrthnd(
  x,
  digits = NULL,
  decimal_mark = NULL,
  big_mark = NULL,
  .pillar = FALSE,
  .trim = TRUE,
  .full_tag = FALSE
)

```

## Arguments

x	A <a href="#">shrthnd_num()</a> vector
digits	Number of digits to apply to shrthnd_double vectors
decimal_mark	The symbol used for decimal marks
big_mark	The symbol used to separate large numbers
.pillar	A flag for formatting within the {pillar} package
.trim	A flag to remove formatting white space
.full_tag	A flag to display full shrthnd tag information

## Details

When calling `as.character()` on a `shrthnd_num()` the output is as you would expect when calling it on a traditional numeric vector, `as_shrthnd()` returns a character vector combining the numeric vector and the shorthand tags.

When `digits = NULL` then `shrthnd_double` vectors are printed with the number of digits set in the `digits` attribute of the vector, setting `digits` in `as_shrthnd()` will override this value.

`as_shrthnd()` acts as the underlying formatter for the printing of `shrthnd_num()` vectors to the console, including inside of `data.frame()` and `tibble::tibble()` objects. When called directly `as_shrthnd()` trims formatting whitespace, set `.trim = FALSE` to return a character vector including formatting whitespace. For ease of display, tags are limited to three characters and replaced with an ellipsis (...) if longer, set `.full_tag = FALSE` to show the entire tag in the output vector.

## Value

A character vector

## See Also

[is\\_shrthnd\\_num\(\)](#), [make\\_shrthnd\\_num\(\)](#), [shrthnd\\_coercion](#), [shrthnd\\_maths](#), [shrthnd\\_num\(\)](#), [shrthnd\\_tags\(\)](#), [tag\\_match\(\)](#)

## Examples

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"), digits = 1)
as_shrthnd(sh_x)
as_shrthnd(sh_x, digits = 3)
```

---

find\_annotations

*Find annotations in a data frame*

---

## Description

`find_annotations()` takes a data frame and identifies possible annotations contained within it and returns them as a named list. `guess_annotations()` is a low-level helper that extracts annotations and returns them as a tibble of cell values, row and column positions.

## Usage

```
find_annotations(
  df,
  type = c("sheet", "cells"),
  title_first = TRUE,
  guess_source = TRUE,
  .row_var = row,
  .col_var = col,
  .value_var = value
```

```

)

guess_annotations(
  df,
  type = c("sheet", "cells"),
  .row_var = row,
  .col_var = col,
  .value_var = value
)

```

### Arguments

df	A data frame object
type	Whether the data frame is in "sheet" format or "cells" format
title_first	Whether the first annotation should be treated as the table title
guess_source	Whether to guess a source note from the annotations
.row_var	When using type = "cells" the name of the variable with row positions
.col_var	When using type = "cells" the name of the variable with column positions
.value_var	When using type = "cells" the name of the variable with row positions

### Details

Data frames have a declared type, which must be either "sheet" format (the default) or "cells" format. "sheet" format is a standard two-dimensional data frame format, such as those read in by `base::read.csv()` or `readxl::read_excel()`. "cells" format is for data frames where each row represents a cell from a spreadsheet and contains a variable for the cell's value, and separate variables providing the row and column variable.

By default `find_annotations()` will try to help parse the annotations found by `guess_annotations()`. With `title_first = TRUE`, the first annotation found in a data frame is assumed to provide a title or label for the table contained in the data frame. With `guess_source = TRUE`, the annotations will be searched for one starting with either "Source:", "Data source:" or "Source data:".

When using type = "cells" the variables identifying the row, column and cell values are specified by `.row_var`, `.col_var` and `.value_var` respectively.

### Examples

```

example_df <- tibble::tibble(
  col1 = c(
    "Table 1", "An example sheet", "species", "Adelie", "Gentoo", "Chinstrap",
    "This table is based on data in the palmerpenguins R package",
    "Source: {palmerpenguins} R package"
  ),
  col2 = c(NA_character_, NA_character_, "bill_length_mm", "38.791",
    "47.505", "48.834", NA_character_, NA_character_),
  col3 = c(NA_character_, NA_character_, "bill_depth_mm", "18.346",
    "14.982", "18.421", NA_character_, NA_character_)
)

```

```
example_df  
  
find_annotations(example_df)  
  
guess_annotations(example_df)
```

---

is_shrthnd_num	<i>Test if an object is using shrthnd</i>
----------------	---

---

### Description

The `is_shrthnd_*` family of functions test whether a vector is either a `shrthnd_num()`, or a `shrthnd_list()`. `is_shrthnd_integer()` and `is_shrthnd_double()` test whether an object is a `shrthnd_num()` vector and whether the underlying data type is an `integer()` or a `double()`. `is_shrthnd_tbl()` test whether an object is a `shrthnd_tbl()` tibble.

### Usage

```
is_shrthnd_num(x)  
  
is_shrthnd_integer(x)  
  
is_shrthnd_double(x)  
  
is_numeric(x)  
  
is_shrthnd_list(x)  
  
is_shrthnd_tbl(x)
```

### Arguments

x	An object to be tested
---	------------------------

### Details

In keeping with base R practice around complex numeric objects such as `Date()`, `difftime()` and `POSIXct()`, using `is_numeric()` on a `shrthnd_num()` vector will return `FALSE`. The `is_numeric()` function included in `{shrthnd}` will return `TRUE` if a vector is either a standard numeric vector or is a `shrthnd_num()`.

### Value

A logical vector

**See Also**

[as\\_shrthnd\(\)](#), [make\\_shrthnd\\_num\(\)](#), [shrthnd\\_coercion](#), [shrthnd\\_maths](#), [shrthnd\\_num\(\)](#), [shrthnd\\_tags\(\)](#), [tag\\_match\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
is_shrthnd_num(sh_x)
is_shrthnd_double(sh_x)

y <- c("12", "34", "[c]", "NA", "56[e]", "78", "90[e]")
sh_y <- shrthnd_num(y, c("[c]", "[e]"))
is_shrthnd_num(sh_y)
is_shrthnd_integer(sh_y)

z <- 1:10
is.numeric(x)
is.numeric(z)
is_numeric(x)
is_numeric(z)

sh_l <- shrthnd_list(sh_x)
is_shrthnd_list(sh_l)

tbl <- tibble::tibble(x = x, sh_x = sh_x)
sh_tbl <- shrthnd_tbl(tbl, title = "Example table")
is_shrthnd_tbl(sh_tbl)
```

---

make\_shrthnd\_num

*Make a shrthnd\_num vector from numeric and character components*

---

**Description**

make\_shrthnd\_num() allows you to construct a shrthnd\_num vector from a numeric vector of data values and a character vector of shorthand markers.

**Usage**

```
make_shrthnd_num(x = numeric(), tags = character(), digits = 2L)
```

**Arguments**

x	A numeric vector
tags	A character vector
digits	The number of digits to format the numeric vector with



**Value**

A shrthnd\_num vector

**See Also**

[as\\_shrthnd\(\)](#), [is\\_shrthnd\\_num\(\)](#), [shrthnd\\_coercion](#), [shrthnd\\_maths](#), [shrthnd\\_num\(\)](#), [shrthnd\\_tags\(\)](#), [tag\\_match\(\)](#)

**Examples**

```
make_shrthnd_num(c(1:3, NA, 4:5, NA), c("", "", "", "[c]", "", "[e]", NA))
```

---

note_to_title	<i>Move notes to and from the title/source note of a tibble</i>
---------------	---

---

**Description**

A shrthnd\_tbl() has three sets of [annotations](#), the note\_to\_\*() functions allow you to move a general note to either the title or source note of a tibble. The \*\_to\_notes() functions do the opposite and (re)insert either the title and/or source note back into the general notes.

**Usage**

```
note_to_title(x, note, .overwrite = FALSE)
```

```
note_to_source_note(x, note, .overwrite = FALSE)
```

```
title_to_notes(x, .add_before = 0)
```

```
source_to_notes(x, .add_before = Inf)
```

```
title_source_to_notes(x, .add_before = 0)
```

**Arguments**

x	A shrthnd_tbl() object
note	The number of the note to move
.overwrite	Whether to overwrite existing
.add_before	Where to (re)insert the note

**Details**

For title\_to\_notes() and title\_source\_to\_notes() the default is to (re)insert the note at the start of the set of notes, for source\_to\_notes() the default is to (re)insert the note at the end of the set of notes.

**Value**

A `sh_rthnd_tbl()`

**See Also**

[annotations\(\)](#), [sh\\_rthnd\\_tbl\(\)](#), [zap\\_sh\\_rthnd\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- sh_rthnd_num(x, c("[c]", "[e]"))
tbl <- tibble::tibble(x = x, sh_x = sh_x)
```

```
sh_tbl <- sh_rthnd_tbl(tbl) |>
  set_notes(c("Note 1", "Note 2", "Note 3")) |>
  note_to_title(1) |>
  note_to_source_note(2)
```

```
sh_tbl
```

```
sh_rthnd_notes(sh_tbl)
```

```
sh_tbl <- sh_tbl |>
  title_to_notes()
```

```
sh_rthnd_notes(sh_tbl)
```

---

sh\_rthnd\_coercion

*Coercion of sh\_rthnd\_num vectors*

---

**Description**

As an extension of the `{vectrs}` package, a `sh_rthnd_num()` is generally coerced to behave as if it was a regular a `numeric()` vector. Where `{vectrs}` doesn't automatically support coercion custom methods are provided to enable a `sh_rthnd_num()` to be considered as a numeric vector.

**General principles**

The principles underpinning the coercion of a `sh_rthnd_num()` vector are that to maximise compatibility with base R and other packages, the vector should generally behave as a numeric vector. This means that `as.numeric()` will produce a bare numeric vector containing just the numeric component of a `sh_rthnd_num()`. Similarly `as.character()` will produce a character vector of the numeric component of a `sh_rthnd_num()`. To work with tags use `sh_rthnd_tags()` and the related [tag location](#) functions. To produce a traditional character vector combining the numeric component and tag component use `as_sh_rthnd()` on a `sh_rthnd_num()` vector.

In keeping with base R practice around complex numeric objects such as `Date()`, `difftime()` and `POSIXct()`, using `is.numeric()` on a `sh_rthnd_num()` vector will return `FALSE`. Use `is_sh_rthnd_num()` to test if a vector is a `sh_rthnd_num()` vector.

See [shorthand\\_maths](#) for details on how `shorthand_num()` works with arithmetic, mathematical and (some) statistical operations.

### Missing values

Of particular note is that using `is.na()` on a `shorthand_num()` vector is designed to work on the numeric component, i.e. if numeric component is missing but a tag marker is present then `is.na()` will return TRUE. Use `is_na_tag()` to identify where there is no tag marker, or `is_na_both()` to identify where both the numeric and tag components are missing.

### See Also

[as\\_shorthand\(\)](#), [is\\_shorthand\\_num\(\)](#), [make\\_shorthand\\_num\(\)](#), [shorthand\\_maths](#), [shorthand\\_num\(\)](#), [shorthand\\_tags\(\)](#), [tag\\_match\(\)](#)

### Examples

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shorthand_num(x, c("[c]", "[e]"))

as.numeric(sh_x)

as.character(sh_x)

is.na(sh_x)
```

---

shorthand\_list

*List the shorthand in a vector*

---

### Description

`shorthand_list()` generates a lookup table of shorthand markers in a vector, either a character vector containing shorthand or a `shorthand_num()` vector.

### Usage

```
shorthand_list(
  x,
  shorthand = NULL,
  na_values = c("", "NA"),
  dec = ".",
  bigmark = ",",
)
```

**Arguments**

x	A character vector containing shorthand, or a shorthand_num() vector
shorthand	A character vector of shorthand values to validate tags against
na_values	A character value of NA values to ignore
dec	The decimal separator for numbers
bigmark	The separator to the left of the decimal separator

**Value**

A list of shorthand positions in a vector

**See Also**

[is\\_shorthand\\_list\(\)](#) [shorthand\\_num\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
shorthand_list(x)

sh_x <- shorthand_num(x)
sh_x
shorthand_list(sh_x)
```

---

shorthand\_maths

*Arithmetic and mathematical operations*


---

**Description**

Arithmetic and most mathematical operations are supported on the numeric component of shorthand\_num() vectors via the {vctrs} package without having to wrap the vector in as.numeric().

**Details**

You can use all the standard arithmetic infix operators (+, -, /, \*, ^, %, %/, !). See vctrs::vec\_arith() for further details.

Through vctrs::vec\_math() the following generic mathematical operations are supported:

- from the [Summary](#) group generic: prod(), sum(), any(), all().
- from the [Math](#) group generic: abs(), sign(), sqrt(), ceiling(), floor(), trunc(), cummax(), cummin(), cumprod(), cumsum(), log(), log10(), log2(), log1p(), acos(), acosh(), asin(), asinh(), atan(), atanh(), exp(), expm1(), cos(), cosh(), cospi(), sin(), sinh(), sinpi(), tan(), tanh(), tanpi(), gamma(), lgamma(), digamma(), trigamma().
- vctrs::vec\_math() also enables support for mean(), is.nan(), is.finite() and is.infinite().

- In addition to these, the {shorthand} package also provides methods for `range()`, `min()`, `max()`, `median()` and `quantile()`. A `shorthand_num()` will work with `sd()` due to the ability of a `shorthand_num()` to be easily coerced to a numeric vector.

For other operations you will need to wrap the `shorthand_num` vector in `as.numeric()`.

For all operations remember that you will likely need to set `na.rm = TRUE` or whatever other method a function has for ignoring missing values.

### See Also

[as\\_shorthand\(\)](#), [is\\_shorthand\\_num\(\)](#), [make\\_shorthand\\_num\(\)](#), [shorthand\\_coercion](#), [shorthand\\_num\(\)](#), [shorthand\\_tags\(\)](#), [tag\\_match\(\)](#)

### Examples

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shorthand_num(x, c("[c]", "[e]"))

sh_x * 2

2 + sh_x

mean(sh_x, na.rm = TRUE)
```

---

shorthand\_num

*Convert a character vector containing shorthand*

---

### Description

`shorthand_num()` coerces a character vector containing numeric data values with non-numeric tags into a numeric-like vector while also retaining the tags.

### Usage

```
shorthand_num(
  x,
  shorthand = NULL,
  na_values = c("", "NA"),
  digits = 2L,
  paren_nums = c("negative", "strip"),
  dec = ".",
  bigmark = ",",
  convert_percent = TRUE
)
```

**Arguments**

x	A character vector of numeric values with shorthand
shorthand	A character vector of shorthand values
na_values	A character vector of NA values
digits	The number of digits for formatting numbers
paren_nums	How to handle numbers in parenthesis (e.g. (12,435.43)), defaults to negative as most commonly used in accounting to denote negative values instead of a minus symbol preceding the value
dec	The decimal separator for numbers
bigmark	The separator to the left of the decimal separator
convert_percent	Whether to convert percentages into decimals

**Details**

Data stored in documents and publications are regularly annotated with shorthand and symbols. Often these tags are found in the same container (e.g. a table or spreadsheet cell) as the value they are associated with, which requires further cleaning of the vector to extract the numeric values.

A simple approach is to discard the non-numeric components, however these tags can convey information which you may wish to retain. `shrthnd_num()` provides a data type that can store both the numeric data and the marker.

By default `shrthnd_num()` will extract any non-numeric values following numeric ones and process them as a shorthand tag. However, you can optionally supply a vector of tags, using the `shorthand` argument, if you wish to validate the extracted tags and only accept vectors with specific shorthand values.

If the underlying numeric values are real numbers (i.e. a `double()` vector) the `digits` argument will be used to format the display of the `shrthnd_dbl` vector (defaults to 2 decimal places).

**Value**

A `shrthnd_num` vector

**See Also**

[shrthnd\\_tags\(\)](#)

[as\\_shrthnd\(\)](#), [is\\_shrthnd\\_num\(\)](#), [make\\_shrthnd\\_num\(\)](#), [shrthnd\\_coercion](#), [shrthnd\\_maths](#), [shrthnd\\_tags\(\)](#), [tag\\_match\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
shrthnd_num(x, c("[c]", "[e]"))
```

---

shorthand_tags	<i>Get the tags attached to a shorthand vector</i>
----------------	--

---

**Description**

shorthand\_tags() provides a character vector the same length as x with the shorthand tags, or NA if that value has no tag. shorthand\_unique\_tags() is a convenience wrapper for unique(shorthand\_tags(x)), but can also be called on a shorthand\_list() object.

**Usage**

```
shorthand_tags(x)
```

```
shorthand_unique_tags(x)
```

**Arguments**

x                    A shorthand\_num() vector

**Value**

A character vector

**See Also**

[as\\_shorthand\(\)](#), [is\\_shorthand\\_num\(\)](#), [make\\_shorthand\\_num\(\)](#), [shorthand\\_coercion](#), [shorthand\\_maths](#), [shorthand\\_num\(\)](#), [tag\\_match\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shorthand_num(x, c("[c]", "[e]"))
shorthand_tags(sh_x)
shorthand_unique_tags(sh_x)
```

---

shorthand_tbl	<i>Add annotations to tibbles</i>
---------------	-----------------------------------

---

**Description**

shorthand\_tbl() provides a way to attach annotations to a table. Specifically, it supports three types of annotation: a title, a source note and general notes. The title and source\_note are each character vectors of length 1, while notes can be a character vector of any length.

**Usage**

```
shorthand_tbl(tbl, title = NULL, notes = NULL, source_note = NULL)
```

**Arguments**

tbl	A <code>tibble::tibble()</code> or object that can be coerced to a tibble.
title	A character vector for the title of tbl
notes	A character vector of general notes relating to tbl
source_note	A character vector for a source note relating to tbl

**Value**

A tibble with `shrthnd` annotations

**See Also**

[is\\_shrthnd\\_tbl\(\)](#)  
[annotations\(\)](#), [note\\_to\\_title\(\)](#), [zap\\_shrthnd\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
tbl <- tibble::tibble(x = x, sh_x = sh_x)
shrthnd_tbl(
  tbl,
  title = "Example table",
  notes = c("Note 1", "Note 2"),
  source_note = "Shrthnd documentation, 2023"
)
```

---

tag\_match

*Get tag locations*


---

**Description**

Base R's matching and location functions will work directly with the numeric component of a `shrthnd_num()` vector, these functions provide the same functionality but applied to the tag component.

**Usage**

```
tag_match(x, tag)
```

```
tag_in(x, tag)
```

```
where_tag(x, tag)
```

```
any_tag(x)
```



```

is_na_tag(x)

is_na_both(x)

locate_tag(x, tag)

locate_any_tag(x)

locate_no_tag(x)

```

### Arguments

x	A shrthnd_num() vector
tag	A single tag to locate

### Details

tag\_match() and tag\_in() are wrappers around vctrs::vec\_match() and vctrs::vec\_in() and thus equivalent to match() and %in% as applied to the tag components of a shrthnd\_num(). tag\_match() will return an integer vector showing the first location of the tag provided, tag\_in() will return TRUE or FALSE depending on whether the tag is in the vector's shorthand.

where\_tag() is equivalent to computing tags == tag, any\_tag() is equivalent to !is.na(tags). Using is.na() on a shrthnd\_num() will assess if the numeric component is missing, is\_na\_tag() is equivalent to is.na(tags), is\_na\_both() tests if both the numeric and tag components of a shrthnd\_num() are missing. They return a logical vector the same length as x.

locate\_tag(), locate\_any\_tag(), located\_no\_tag() are equivalent to passing the return values of where\_tag(), any\_tag() and is\_na\_tag() to which(). They return an integer vector the same length as x.

### Value

For tag\_match(), locate\_tag(), locate\_any\_tag() and locate\_no\_tag() an integer vector. For tag\_in(), where\_tag(), any\_tag(), is\_na\_tag() and is\_na\_both() a logical vector.

### See Also

[as\\_shrthnd\(\)](#), [is\\_shrthnd\\_num\(\)](#), [make\\_shrthnd\\_num\(\)](#), [shrthnd\\_coercion](#), [shrthnd\\_maths](#), [shrthnd\\_num\(\)](#), [shrthnd\\_tags\(\)](#)

### Examples

```

x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
shrthnd_tags(sh_x)

tag_match(sh_x, "[e]")

tag_in(sh_x, "[e]")

```

```
where_tag(sh_x, "[e]")
any_tag(sh_x)
is_na_tag(sh_x)
is_na_both(sh_x)
locate_tag(sh_x, "[e]")
locate_any_tag(sh_x)
locate_no_tag(sh_x)
```

---

where\_shrthnd\_cols      *Identify which columns use shrthnd*

---

### Description

where\_shrthnd\_cols() applies is\_shrthnd\_num() across columns of a data.frame (or elements in a list). which\_shrthnd\_cols() identifies the columns in a data.frame or (elements of a list) by name or index position. any\_shrthnd\_cols() tests whether a data.frame (or list) has any columns that are shrthnd\_num() vectors.

### Usage

```
where_shrthnd_cols(x)
which_shrthnd_cols(x, .names = FALSE)
any_shrthnd_cols(x)
```

### Arguments

x	A data.frame (or list)
.names	A logical vector indicating whether to return column names or an integer vector of column positions (the default)

### Value

For where\_shrthnd\_cols() a logical vector of the same length as the number columns in x. For which\_shrthnd\_cols() a character vector of names or an integer vector of index positions (the default). For any\_shrthnd\_cols() either TRUE if there are any shrthnd\_num() vectors in the object or FALSE if not.

### See Also

[shrthnd\\_num\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
tbl <- tibble::tibble(x = x, sh_x = sh_x)
```

```
where_shrthnd_cols(tbl)
```

```
which_shrthnd_cols(tbl)
```

```
which_shrthnd_cols(tbl, .names = TRUE)
```

```
any_shrthnd_cols(tbl)
```

---

zap_shrthnd	<i>Remove annotations from tibbles</i>
-------------	--

---

**Description**

The zap\_\*() functions remove annotations from a shrthnd\_tbl() object. zap\_title(), zap\_source\_note() and zap\_notes() remove the title, source note and general notes respectively. zap\_tbl() removes all three types of annotations and also strips the shrthnd\_tbl class from the object. zap\_shrthnd() is a low-level helper function that power the attribute removal.

**Usage**

```
zap_title(x)
```

```
zap_source_note(x)
```

```
zap_notes(x)
```

```
zap_tbl(x)
```

```
zap_shrthnd(x, what = c("title", "source_note", "notes"), zap_class = FALSE)
```

**Arguments**

x	A shrthnd_tbl()
what	One or more of title, source_note or notes indicating which set of notes to remove
zap_class	Whether to remove the "shrthnd_tbl" class

**Details**

To remove shrthnd from a shrthnd\_num() vector use as.numeric(), as.character() or as\_shrthnd() to coerce the vector to another type.

**Value**

Returns `x` with relevant attributes removed

**See Also**

[annotations\(\)](#), [note\\_to\\_title\(\)](#), [shrthnd\\_tbl\(\)](#)

**Examples**

```
x <- c("12", "34.567", "[c]", "NA", "56.78[e]", "78.9", "90.123[e]")
sh_x <- shrthnd_num(x, c("[c]", "[e]"))
tbl <- tibble::tibble(x = x, sh_x = sh_x)
sh_tbl <- shrthnd_tbl(
  tbl,
  title = "Example table",
  notes = c("Note 1", "Note 2"),
  source_note = "Shrthnd documentation, 2023"
)

sh_tbl

zap_title(sh_tbl)

zap_source_note(sh_tbl)

zap_notes(sh_tbl)

zap_tbl(sh_tbl)
```

# Index

- \* **num**
  - as\_shrthnd, 4
  - is\_shrthnd\_num, 7
  - make\_shrthnd\_num, 8
  - shrthnd\_coercion, 10
  - shrthnd\_maths, 12
  - shrthnd\_num, 13
  - shrthnd\_tags, 15
  - tag\_match, 16
- \* **tbl**
  - annotations, 2
  - note\_to\_title, 9
  - shrthnd\_tbl, 15
  - zap\_shrthnd, 19
- add\_notes (annotations), 2
- add\_notes<- (annotations), 2
- annotations, 2, 9, 10, 16, 20
- any\_shrthnd\_cols (where\_shrthnd\_cols), 18
- any\_tag (tag\_match), 16
- as\_shrthnd, 4, 8, 9, 11, 13–15, 17
- find\_annotations, 5
- guess\_annotations (find\_annotations), 5
- is\_na\_both (tag\_match), 16
- is\_na\_tag (tag\_match), 16
- is\_numeric (is\_shrthnd\_num), 7
- is\_shrthnd (is\_shrthnd\_num), 7
- is\_shrthnd\_double (is\_shrthnd\_num), 7
- is\_shrthnd\_integer (is\_shrthnd\_num), 7
- is\_shrthnd\_list (is\_shrthnd\_num), 7
- is\_shrthnd\_list(), 12
- is\_shrthnd\_num, 5, 7, 9, 11, 13–15, 17
- is\_shrthnd\_tbl (is\_shrthnd\_num), 7
- is\_shrthnd\_tbl(), 16
- locate\_any\_tag (tag\_match), 16
- locate\_no\_tag (tag\_match), 16
- locate\_tag (tag\_match), 16
- make\_shrthnd\_num, 5, 8, 8, 11, 13–15, 17
- Math, 12
- note\_to\_source\_note (note\_to\_title), 9
- note\_to\_title, 3, 9, 16, 20
- set\_notes (annotations), 2
- set\_source\_note (annotations), 2
- set\_tbl\_antn (annotations), 2
- set\_title (annotations), 2
- shrthnd\_coercion, 5, 8, 9, 10, 13–15, 17
- shrthnd\_list, 11
- shrthnd\_maths, 5, 8, 9, 11, 12, 14, 15, 17
- shrthnd\_notes (annotations), 2
- shrthnd\_notes<- (annotations), 2
- shrthnd\_num, 5, 8, 9, 11, 13, 13, 15, 17
- shrthnd\_num(), 4, 12, 18
- shrthnd\_source\_note (annotations), 2
- shrthnd\_source\_note<- (annotations), 2
- shrthnd\_tags, 5, 8, 9, 11, 13, 14, 15, 17
- shrthnd\_tags(), 14
- shrthnd\_tbl, 3, 10, 15, 20
- shrthnd\_title (annotations), 2
- shrthnd\_title<- (annotations), 2
- shrthnd\_unique\_tags (shrthnd\_tags), 15
- source\_to\_notes (note\_to\_title), 9
- Summary, 12
- tag location, 10
- tag\_in (tag\_match), 16
- tag\_match, 5, 8, 9, 11, 13–15, 16
- title\_source\_to\_notes (note\_to\_title), 9
- title\_to\_notes (note\_to\_title), 9
- where\_shrthnd\_cols, 18
- where\_tag (tag\_match), 16
- which\_shrthnd\_cols (where\_shrthnd\_cols), 18

zap\_notes (zap\_shrthnd), [19](#)  
zap\_shrthnd, [3](#), [10](#), [16](#), [19](#)  
zap\_source\_note (zap\_shrthnd), [19](#)  
zap\_tbl (zap\_shrthnd), [19](#)  
zap\_title (zap\_shrthnd), [19](#)